How Does Social Media Misinformation Affect Public Sentiment?

ZHENG WENSHENG, 2021147579
Fall 2024, Computer programming final project

1. Data Preprocessing

The dataset used in this project is derived from the LIAR dataset and Sentiment140, both of which are crucial for analyzing how misinformation impacts public sentiment. Preprocessing steps ensured the data was compatible for identifying patterns in misinformation and its emotional effects. The following processing steps were applied:

- **LIAR Dataset:** Extracted key fields such as "Label" and "Statement." Each statement was assigned a sentiment score using the Vader SentimentIntensityAnalyzer. Additional features such as text length and word count were generated to enrich the dataset used by Naive Bayes Model and Random Forest Classifier.
- **Sentiment140 Dataset:** Polarity values were mapped to "Positive" and "Negative." Text was preprocessed by tokenization and removal of unnecessary symbols.

LIAR-Analysis Schema	
Label	Veracity label for the statement.
Statement	The text content of the claim or statement.
Sentiment	The sentiment analysis result.
Word Count	The total word count of the statement.

Table 1: Preprocessed LIAR Dataset Schema

Sentiment140-Analysis Schema	
Polarity	The sentiment polarity of the tweet.
Text	The text content of the tweet.
Sentiment	The sentiment analysis result after processing.

Table 2: Preprocessed Sentiment140 Dataset Schema

Processed datasets were saved as CSV files for subsequent analysis:

- `liar_sentiment_analysis.csv`
- `sentiment140_analysis.csv`

The preprocessing steps were essential for ensuring the datasets were suitable for the analysis goals of this project:

- **Consistency Across Datasets:** Since the LIAR dataset and Sentiment140 dataset originated from different sources, preprocessing was necessary to harmonize their formats.
- **Feature Extraction:** Adding features such as `Word Count` and `Sentiment` enabled deeper insights into the relationship between misinformation and virality, as well as the emotional impact of misinformation.
- **Focus on Relevant Fields:** By selecting only essential columns, such as `Label`, `Statement`, and `Polarity`, unnecessary noise was removed, allowing the analysis to focus on the project's core objectives.

These preprocessing steps transformed the raw datasets into well-structured, analyzable formats, paving the way for meaningful sentiment analysis and machine learning applications.

2. Problem-Solving Approach (Algorithm Overview)

2.1 Overview of Methods

To address the hypothesis that misinformation on social media influences public sentiment, the following methods were employed:

- **Naive Bayes Classifier:** A probabilistic model was used for misinformation detection in the LIAR-Analysis dataset. By transforming textual data into feature vectors using *CountVectorizer*, the classifier predicted the veracity of statements based on their textual features.
- **Random Forest Classifier:** A decision-tree-based ensemble model was trained to classify statements as `True` or `False`, leveraging engineered features such as *Sentiment Scores*, *Word Counts*, and *Text Length*.
- **Vader SentimentIntensityAnalyzer:** A lexicon-based sentiment analysis tool was applied to both datasets to compute sentiment scores, facilitating the analysis of public sentiment associated with misinformation.

2.2 Key Steps in the Algorithm

The key steps in the algorithms and processing pipeline are outlined below:

2.2.1 Naive Bayes Classifier for Veracity Prediction

1. Transformed the `Statement` column into feature vectors using `CountVectorizer`, excluding stop words to reduce noise.
2. Split the data into training and testing sets using an 80-20 ratio.
3. Trained a `MultinomialNB` model on the training set and evaluated it on the testing set.
4. Achieved an accuracy of approximately 26%, highlighting the inherent difficulty of the task and the imbalanced nature of the dataset.

Main algorithm 1: Naive Bayes Classifier

```
# Transforming statements into feature vectors
vectorizer = CountVectorizer(stop_words='english')
X = vectorizer.fit_transform(liar_data['Statement'])
y = liar_data['Label']

# Splitting data into training and test sets
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size=0.2, random_state=42)

# Training the Naive Bayes model
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)

# Evaluation
y_pred = nb_model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

2.2.2 Random Forest Classifier for Veracity Prediction

1. Engineered features (`Text Length`, `Word Count`, `Sentiment Score`) were used as input.
2. Split the data into training and testing sets using an 80-20 ratio.
3. Trained a `RandomForestClassifier` model and evaluated its performance.
4. The feature importance scores were analyzed to understand the relative contribution of each feature.

Main algorithm 2: Random Forest Classifier

```
# Feature engineering
liar_data['Text Length'] = liar_data['Statement'].apply(len)
X = liar_data[['Text Length', 'Word Count', 'Sentiment Score']]
y = liar_data['Label'].apply(lambda x: 1 if x in ['false', 'pants-fire'] else 0)

# Splitting data into training and test sets
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size=0.2, random_state=42)

# Training the Random Forest model
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

# Evaluation
y_pred = rf_model.predict(X_test)
print("Classification Report:\n", classification_report(y_test, y_pred))
```

2.2.3 Sentiment Analysis with Vader

1. Applied Vader SentimentIntensityAnalyzer to compute sentiment scores for each statement in the LIAR dataset and each tweet in the Sentiment140 dataset.

2. Sentiment results (`Positive` or `Negative`) were used to analyze the relationship between misinformation and public sentiment.

Main algorithm 3: Sentiment Analysis

```
# Function to analyze sentiment
def analyze_sentiment(text):
    sentiment_score = sia.polarity_scores(text)
    return 'Positive' if sentiment_score['compound'] > 0 else 'Negative'

# Applying sentiment analysis
liar_data['Sentiment'] = liar_data['Statement'].apply(analyze_sentiment)
sentiment_data['Sentiment'] = sentiment_data['Text'].apply(analyze_sentiment)
```

2.3 Purpose and Significance of Methods

In summary, we highlight the characteristics of misinformation and demonstrate its profound emotional impact. Through the integration of sentiment analysis and machine learning models, we achieved a multi-faceted exploration of how misinformation propagates and influences public discourse.

3. Python File Structure

3.1 `analysis_module.py`

Core functionalities for data processing, sentiment analysis, and machine learning models.

- **Data Loading**: Functions for loading datasets.
- **Sentiment Analysis**: Functions using the Vader `SentimentIntensityAnalyzer` to classify sentiment.
- **Feature Engineering**: Adding features such as text length and word count.
- **Model Training and Evaluation**: Training Naive Bayes and Random Forest classifiers, along with evaluation metrics.

3.2 `analysis_visualization.py`

Creating visual representations of data analysis results.

- **Sentiment Distribution**: Bar charts showing sentiment distribution across categories.
- **Word Count Analysis**: Visualizations of average word count by sentiment or label.

- **Interactive Features**: Gridline customization and layout enhancements for clearer insights.

3.3 `sentiment_classification.py`

Sentiment classification based on sentiment scores and text length.

- **Sentiment Categorization**: Mapping sentiment scores into positive, neutral, and negative categories.
- **Text Length Analysis**: Evaluates the relationship between sentiment categories and the average length of statements.

3.4 `topic_analysis.py`

Specializes in topic modeling and text feature extraction.

- **Topic Modeling**: Uses Latent Dirichlet Allocation (LDA) to identify topics in text data.
- **N-gram Analysis**: Supports bi-gram and tri-gram feature extraction.
- **Keyword Extraction**: Extracts and displays the top keywords for each identified topic.

3.5 `topic_sentiment_analysis.py`

Combines topic modeling with sentiment analysis to gain deeper insights.

- **Sentiment Distribution by Topic**: Analyzes how sentiment varies across topics.
- **False Statement Analysis**: Evaluates the proportion of false statements for each topic.
- **Sentiment and Label Analysis**: Explores average sentiment scores across different labels and topics.
- **Comparison of Topic Distributions**: Compares how topics differ between false and true statements.

The full codebase and configuration details can be accessed on GitHub:
`https://github.com/Zwensheng01/Final_project`

4. Data Analysis Results

4.1 Keyword Analysis

Topic	Keywords
Topic 1	state budget, 10 percent, african american, federal government
Topic 2	500 million, 40 percent, ted cruz, million jobs, illegal immigrants
Topic 3	says hillary, clinton, says hillary, health care reform, care reform
Topic 4	new york, income tax, president obama, says donald trump
Topic 5	says president, president barack, obamas president, barack obama

Table 3: Top Keywords by Topic

The keyword analysis identifies key themes within misinformation and related topics:

- **Topic 1**: Highlights discussions around state and federal policies.
- **Topic 2**: Centers on economic issues, employment, and financial policies.
- **Topic 3**: Highlight discussions on healthcare policies and political leadership.
- **Topic 4**: Point to statements about taxation policies.
- **Topic 5**: Suggest this topic revolves around legal rulings and healthcare policies.

4.2 Sentiment Distribution by Label/Topic

Label	Negative (%)	Positive (%)
Barely-True	69.62	30.38
False	61.60	38.40
Half-True	68.95	31.05
Mostly-True	66.14	33.86
Pants-Fire	67.24	32.76
True	59.76	40.24

Table 4: SD Across Labels

Topic	Negative (%)	Neutral (%)	Positive (%)
1	36.82	35.21	27.97
2	34.78	35.87	29.35
3	27.01	20.38	52.61
4	39.09	29.44	31.47
5	31.79	34.36	33.85

Table 5: SD Across Topics

4.3 Average Sentiment by Label and Topic

Topic	Barely-True	False	Half-True	Mostly-True	Pants-Fire	True
1	-0.1053	-0.0093	-0.1124	-0.0877	-0.0330	-0.0063
2	-0.0536	-0.0612	-0.0191	0.0956	-0.0716	-0.0442
3	-0.0434	0.2561	0.0447	0.2047	0.0797	0.1342
4	0.0026	-0.0537	-0.0607	-0.1158	-0.2386	0.1625
5	0.0792	0.0856	-0.0679	-0.0205	-0.1702	0.0637

Table 6: Average Sentiment by Label and Topic

From the average sentiment analysis:

- **Topic 3** has the most positive average sentiment overall, particularly for labels like **False** (0.2561) and **Mostly-True** (0.2047), suggesting these statements might be associated with optimistic discussions.
- **Topic 4** displays strong negative sentiment for labels like **Pants-Fire** (-0.2386), reflecting the public's mistrust and emotional reaction to highly deceptive misinformation in taxation-related statements.
- Labels such as **True** and **Barely-True** maintain mixed sentiment across topics, reflecting variability in the statements' credibility.

5. Summary

This study explores the impact of social media misinformation on public sentiment, emphasizing how deceptive statements influence emotional responses across various labels and topics. The analysis highlights that misinformation-associated labels, such as **Pants-Fire** and **False**, are linked to more negative sentiment, with **Pants-Fire** displaying a significant average negative sentiment of -0.2386 in **Topic 4** (taxation policies). In contrast, labels like **True** and **Mostly-True** are associated with higher positive sentiment, particularly in **Topic 3** (healthcare reforms), where a positive sentiment average of 0.2047 was observed. These findings reveal that misinformation exacerbates public distrust and emotional negativity, especially in controversial topics like taxation. Conversely, accurate statements foster greater optimism in socially critical areas, such as healthcare. This underscores the importance of identifying and addressing misinformation to preserve trust and promote constructive public discourse.

6. References

[1] William Yang Wang, "Liar, Liar Pants on Fire: A New Benchmark Dataset for Fake News Detection," Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, 2017.
[2] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," CS224N Project Report, Stanford, 2009.